モンテカルロ法を用いたボードゲームの対戦プログラム : 双対グラフ上の高速プレイアウト手法の提案 (計算理論とアルゴリズムの新潮流)

AUTHOR(S):

神野, 雅俊

CITATION:

神野, 雅俊. モンテカルロ法を用いたボードゲームの対戦プログラム : 双対グラフ上の高速プレイアウト手法の提案 (計算理論とアルゴリズムの新潮流). 数理解析研究所講究録 2014, 1894: 89-94

ISSUE DATE:

2014-05

URL:

http://hdl.handle.net/2433/195833

数理解析研究所講究録
第1894巻 2014年 89-94

モンテカルロ法を用いたボードゲームの対戦プログラム —双対グラフ上の高速プレイアウト手法の提案—

中央大学大学院理工学研究科　神野雅俊
Masatoshi Jinno
Faculty of Science and Engineering, Chuo University

1　はじめに

ボードゲームは古くから多くの人に遊ばれ，今もなお老若男女に親しみがある．現在有名なボードゲームとして囲碁，チェス，オセロなどが挙げられ，人と人に限らず，人とコンピュータが対局できるソフトウェアも多く開発されている．近年は，強いソフトウェアを製作するために，様々なアルゴリズムが研究され，その強さは日進月歩の勢いである．記憶に新しい人間とソフトウェアとの対局として，2013年3月から4月にかけて行われた「第2回将棋　電王戦」がある．この将棋棋戦は，プロ棋士5人とコンピュータソフト5本が対局する団体戦であり，コンピュータは3勝1敗1分けという好成績を残している．

本研究は，ボードゲームHexに対するモンテカルロ法をベースとしたアルゴリズムに用いる新たな高速プレイアウト手法を提案する．提案するプレイアウト手法は，盤面を双対グラフとして捉えたデータ構造を利用し，勝利に必要なパスの探索を行うことで高速化を図っている．

2　Hex

2.1　Hexのルール

Hexは図1のような六角形のセルが敷き詰められた盤面で2人で遊ぶゲームである．両プレイヤーは各々の色を持ち，先手は黒，後手は白となる．盤面上の何も置かれていないセルに，交互に各々の色の石を打ち，先手は盤面の左上と右下を，後手は盤面の右上と左下を繋げることで勝利となる．図2は白の勝利したゲームである．

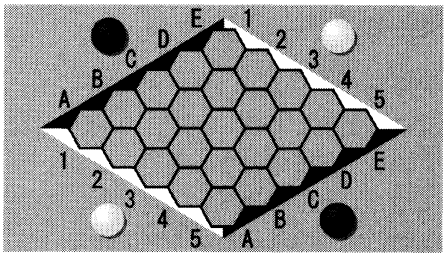

図1: 5 × 5の盤面

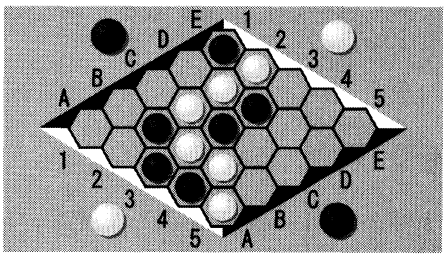

図2: 白の勝利したゲーム

図1，2は，盤面のサイズが5 × 5のHexである．本論文では，記述の無い限り，盤面のサイズを11 × 11とする．これは，国際コンピュータゲーム協会 (International Computer Games Association; ICGA) が主催するComputer Olympiadの，ボード

ゲームプログラム同士を戦わせる世界大会「ICGAトーナメント」のルールに基づいている．

2.2 Hexの特徴

Hexは，デンマークの数学者 Piet Hein によって1942年に考案されたゲームである．また，Piet Heinとは別に，アメリカの数学者 John Nash が1948年に Nash という名前で同じゲームを独立に考案した．その後，科学雑誌に取り上げられたことをきっかけに有名になり，現在もなお研究されている．

Hexには以下の特徴がある，

- 引き分けが存在しない，
- 先手必勝である，
- 必勝手順は不明である．

先手必勝であることは，Nash が Strategy stealing の手法を用いて証明した．しかしこの手法では，先手必勝のゲームであることを証明できるが，その手順については手がかりが得られない．Hexの必勝手順については，今もなお詳細は不明である．

Hexは先手必勝であるため，パイルールという，先手と後手の優劣の差を小さくするためのルールを2.1節のルールに付与することがあるが，本発表にはこのルールを適用しない．

3 モンテカルロ法

ボードゲーム AI に適用されるモンテカルロ法は，評価関数を必要とせず，囲碁などのボードゲームで多大な成果をあげている手法であり，Hexにおいてもそれは例外ではない．本予稿では，ボードゲーム AI に適用されるモンテカルロ法について記述する．

3.1 原始モンテカルロ法

原始モンテカルロ法は，評価関数の代わりに，「ある局面からランダムに手を指し続けたときの勝率を計算し，勝率が高い局面ほど優れた局面と判断する」手法である．つまり，ある局面を評価しようとしたとき，その局面からランダムに手を指し続ける．この，『ランダムに手を指し続けること』をランダムゲームミュレーションと呼び，ランダムゲームシミュレーションを続けて終了局面まで指すことをプレイアウトと呼ぶ．プレイアウトの結果，手番側が勝ちか負けかというゲームの最終結果が得られる．これを難度も繰り返すことで，勝った割合，つまり勝率を得ることができる．この勝率が一番高い局面こそが最も勝ちやすい優れた局面であると考える．

図3は原始モンテカルロ法のゲーム木のモデル図である．全ての候補手に対して同数のプレイアウトを行う．

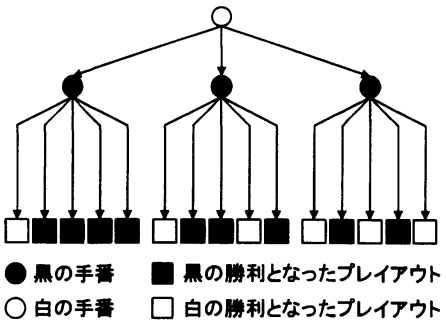

図 3: 原始モンテカルロ法のゲーム木のモデル

3.2 Upper Confidence Bound(UCB)

原始モンテカルロ法には，次の2つの問題点がある．

問題1 明らかな悪手に対してもランダムゲームシミュレーションを均等に実行する．

問題2 平均勝率で判断するため，ランダムゲームシミュレーション内で最悪の手の影響を正確に反映できない．

3.2の問題は，悪手であっても好手と同じ回数のプレイアウトを行うことである．悪手に対して行うランダムゲームシミュレーションを省き，より重要な

手に対してランダムゲームシミュレーションを行うことで，比較的好手の中で，最も勝率の高い手を選択することができる．

この問題を解決する手法の 1 つに，Upper Confidence Bound (UCB) が挙げられる．UCB は，各候補手に対する評価値，UCB 値と呼ばれる値，の高い手を選んでプレイアウトを行う手法である．好手は勝率が高くなるが，プレイアウト回数が少ない場合の勝率の値は信頼性が薄いため，そのような候補手は優先的にプレイアウトを行う必要がある．UCB 値の計算式はいくつか提案されているが，本研究では

$$\mathrm{UCB}(i) = \overline{X}_i + \sqrt{\frac{2\log N}{n_i}} \qquad (1)$$

を適用している．ただし，$\overline{X}_i$ は手 i を指したときの現在の勝率，n_i は手 i 以降をランダムゲームシミュレーションした回数，N はその局面において思考したランダムゲームシミュレーショの総数 (n_i の総和) である．この式を各候補手に適用することで，勝率が高く，プレイアウト回数が少ない手を選んでプレイアウトを行うことができる．

図 4 は，UCB 値を用いたゲーム木のモデル図である．優秀な候補手はプレイアウトする回数が多い．

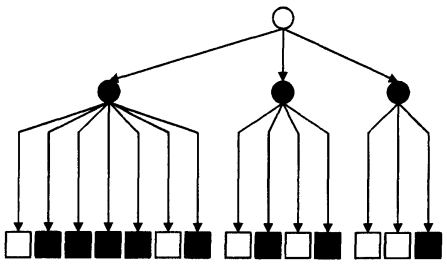

図 4: UCB 値を用いたゲーム木のモデル

3.3 Upper Confidence bound applied to Trees (UCT)

UCB の手法を用いることによって，3.2 節の問題 3.2 は解決することができるが，問題 3.2 は解決できない．この問題を解消できる代表的な手法に Upper Confidence bound applied to Trees (UCT) がある．原始モンテカルロ法，UCB はゲーム木の探索を行っていないため，最小最大戦略では不利と判断される局面においても，これを有利と判断してしまうことがある．UCT は，UCB を用いてゲーム木の探索を行うことで最小最大戦略に近い形にできる．しかし，ゲーム木全体を保持することは現実的に不可能であるため，UCT ではノードが保持する局面から行ったプレイアウト数が閾値を超えた場合，1 手先を展開する．

図 5，6 は，UCT のゲーム木のモデル図になる．図 5 の左のノードを展開すると，図 6 のようになる．

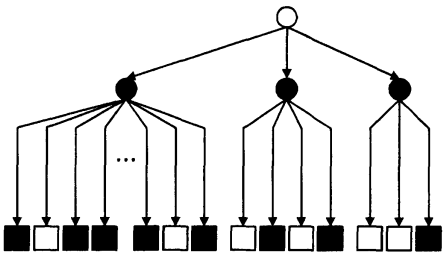

図 5: UCT のゲーム木のモデル．左のノードが閾値に達したため展開する．

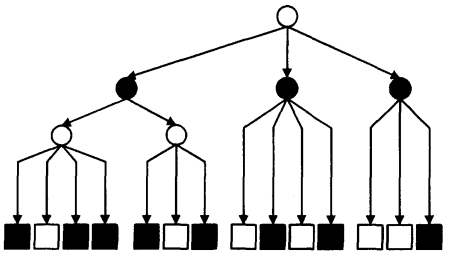

図 6: UCT のゲーム木のモデル．左のノードは展開された

3.4 All Moves As First (AMAF)

シミュレーションの精度は，プレイアウト数に大きく影響され，プレイアウト数が多いほど精度は高

くなることが知られている．しかし，プレイアウトの回数を増やすと計算時間が増えてしまう．そこで，All Moves As First (AMAF) が考案された．AMAF は，1 度のプレイアウトの結果を，木探索で辿ったノード以外に，ランダムゲームシミュレーションの終局盤面に到達できるノードにも反映させる手法である．

例えば，図 7 の盤面から，黒 C3，白 E1，黒 D2，白 E5 のゲーム木探索を経てランダムゲームシミュレーションを行い，図 8 の盤面になったとする．AMAF を用いていない UCT であれば，白 E5，黒 D2，白 E1，黒 C3 のノードのみ結果を反映するが，AMAF を用いた場合は，これら 4 つのノードの兄弟ノードに図 8 の局面に至るようなノードに対しても結果を反映する．具体的には，黒 D2 の兄弟ノードの 1 つ黒 B2 に指すのノードなどになる．このようにして 1 度のプレイアウトの結果を多くのノードに反映させることで少ないプレイアウト数でゲーム木の展開が早くなり，より高精度な探索が可能になる．

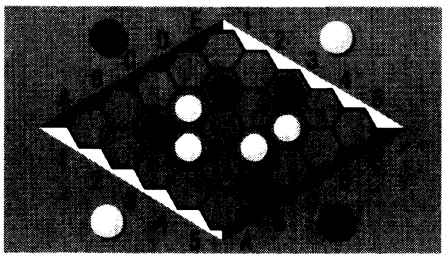

図 7: 現在の盤面

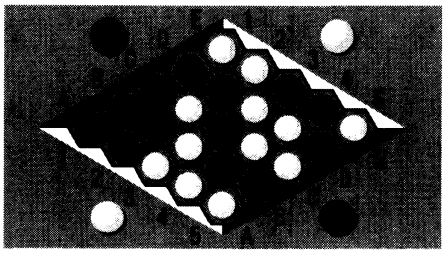

図 8: 図 7 の盤面から，黒 C3，白 E1，黒 D2，白 E5 のゲーム木探索，プレイアウトを行った時点の盤面

4 プレイアウト

本研究は，新たな高速プレイアウト手法を提案する．本章では，Hex において主流とされているプレイアウトと提案する高速プレイアウト手法について記述する．

4.1 既存研究

Hex において主流とされているランダムゲームシミュレーションは，ゲームの進行と同様に行うものである．具体的には，空のセルからランダムに 1 つ選び黒を指す，続いて空のセルからランダムに 1 つ選び白を指す，ということを続ける．しかし，シミュレーションの終了は，どちらかが勝利したときではなく，空のセルがなくなったときとなる．1 手指すごとに終了判定をするよりも，空セル数の回数だけ乱数を生成し，1 度終了判定をする方が平均的な処理時間は短いからである．

4.2 高速プレイアウト手法

既存の主流とされるプレイアウトは，空のセル全てに対して乱数生成を行い，その後勝敗判定をする．このため，勝敗に関係ない処理が多いという問題を内包している．この問題に対して，勝敗判定を行いながら，ランダムゲームシミュレーションを行えるプレイアウト手法を提案し，プレイアウトの高速化を図る．

勝敗判定は，セルとセルの間を辿り盤面のどこに行き着くかを調べる手法をとっている．図 9 は，空セルがなくなりゲームが終了した盤面である．盤面を左端から見て，左側が黒，右側が白となるようにセルとセルの間を縫うように進む．進み続けると，盤面の下もしくは上に抜ける．このとき，下に抜けたときは黒が勝利しており，上に抜けたときは白が勝利している．

これを，ゲームが終了していない盤面に当てはめる．例えば，図 10 の勝敗判定を開始すると，B2 の空セルに到達する．ここで，空セルに 0.5 の確率で

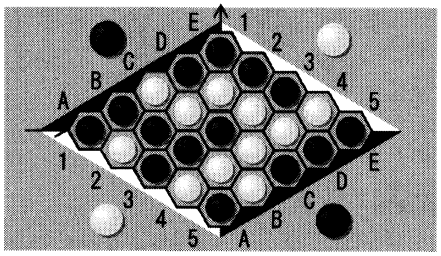

図 9: 勝敗判定方法．上に抜けているので白の勝利と判断できる．

黒を，0.5 の確率で白を置き，その後，勝敗判定を続行する．この手法により，ランダムゲームシミュレーションを従来の手法よりも少ない処理数で行うことができる．

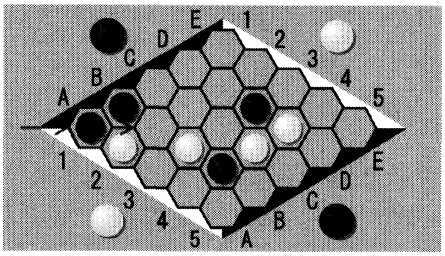

図 10: 高速プレイアウト手法．勝敗判定中に空セルが出現したとき，0.5 の確率で黒もしくは白を置く．

5 計算機実験

5.1 高速プレイアウト手法の計算時間

提案する高速プレイアウト手法が，従来の手法と比べてどの程度高速化されているかを実験に基づき調べる．実験方法は，対局中に出現する盤面に対して，従来のプレイアウトと高速プレイアウト手法を 100 万回実行するのにかかった時間を計測する．ゲーム木探索などの UCT の処理は含まないものとする．なお，対局中に出現する盤面は，Computer Olympiad の 2008 年，2009 年大会の棋譜とした．対局数は 36 局，総盤面数は 2037 である．

図 11 は，従来のプレイアウトと高速プレイアウト手法の計算時間を示す．高速プレイアウト手法の計算時間が従来のプレイアウトの計算時間よりも短いことがわかる．

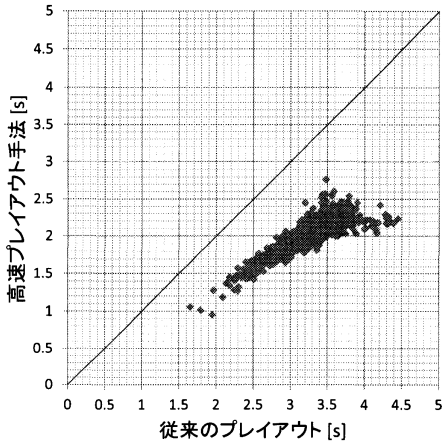

図 11: 従来のプレイアウトと高速プレイアウト手法の計算時間

5.2 高速プレイアウト手法の評価

5.1 節の実験から，従来のプレイアウトに比べ高速プレイアウト手法は計算時間が短くなったことがわかった．次に，この手法がどの程度実用性があるかを調べる．実験方法は，従来のプレイアウトを用いた UCT と高速プレイアウト手法を用いた UCT を 1000 局対局させ，勝率を調べる．次の手を決定するまでの計算時間は 20 秒とする．

実験結果の結果，次のようになった．数値は先手の勝率 [%] を示す．第 1 行，第 1 列の「従来」,「高速」はそれぞれ「従来のプレイアウトを用いた UCT」と「高速プレイアウト手法を用いた UCT」を指す．

従来のプレイアウトによる UCT に比べ，高速プレイアウト手法を用いた UCT は，手番に依らず高い勝率を得ることができることが読み取れる．

表 1: 2 つのプレイアウト手法による UCT の勝率

先手 ＼ 後手	従来	高速
従来	53.2	36.8
高速	74.6	53.1

5.3 高速プレイアウト手法を適用した AMAF

AMAF を適用した UCT の実用性を測る実験を行った．従来のプレイアウトを用いた UCT と高速プレイアウト手法を用いた UCT に対して，AMAF の有無により勝率がどのように変化するかを調べる．それぞれの AI 同士の対局数は 200 局とし，次の手を決定するまでの時間は 20 秒とする．表中の数値は先手の勝率 [%] を示す．第 1 行，第 1 列の「従来」，「従来 AMAF」，「高速」，「高速 AMAF」はそれぞれ「従来のプレイアウトによる UCT」，「AMAF を適用した従来のプレイアウトによる UCT」，「高速プレイアウト手法による UCT」，「AMAF を適用した高速プレイアウト手法による UCT」を示す．括弧内の数値は 5.2 節の実験結果を引用したものである．

表 2: AMAF を適用した UCT を含む 4 つの AI の勝率

先手 ＼ 後手	従来	従来 AMAF	高速	高速 AMAF
従来	(53.2)	20.5	(36.8)	35.0
従来 AMAF	97.0	—	99.5	46.0
高速	(74.6)	17.0	(53.1)	43.5
高速 AMAF	94.0	30.0	91.0	—

AMAF を適用していない UCT との対局結果から，AMAF を適用することで，どちらのプレイアウト手法による UCT でも勝率が上がることがわかる．また，AMAF を適用した UCT が後手であれば，先手が AMAF を適用した UCT であっても勝ち越す結果が得られている．Hex は先手に有利なゲームであるにもかかわらず，なぜこのような結果が得られたのかは不明であるが，試行回数が少ないためではないかと考える．

6 結論

本研究において，UCT を基礎とした Hex の対戦プログラムを構築し，提案する高速プレイアウト手法が従来のプレイアウト手法に比べて有用であることを実験により確かめた．また，高速化することにも成功し，実験に用いた 2037 の盤面全てで従来のプレイアウトよりも高速化できている．AMAF を適用した UCT では，AMAF を適用していない AI に対して勝率を大きく伸ばしている．しかし，AMAF を適用した UCT が対局相手の場合，先手が勝ち越せていないため，さらなる実験と考察が課題となる．

参考文献

[1] 小谷善行, 岸本章宏, 芝原一友, 鈴木豪, 「ゲーム計算メカニズム」, コロナ社, 2010.

[2] B. Arneson, R. Hayward, P. Henderson, "Monte Carlo Tree Search in Hex," *IEEE Transactions on Computational Intelligence and AI in Games*, **2** (2010), pp. 251–257.

[3] Y. Peres, O. Schramm, S. Sheffield, D. B. Wilson, "Random-turn Hex and Other Selection Games," *American Mathematical Monthly*, **114** (2007), pp. 373–387.